WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Monday, December 19, 2005

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L16	L15 and table	15
☐	L15	L14 and @AD<20010816	23
☐	L14	(multi-processor or (multiple processor)) near8 (partition or partitioning or configure or configuring) near8 (algorithm or mechanism or software or schema or program or policy or rule or regulation)	40
☐	L13	L12 and l9	0
☐	L12	L11 and @AD<20010816	21
☐	L11	(dynamic or run-time) near8 (altering or changing or updating or configuring or reconfiguring) near8 (routing table)	36
☐	L10	(dynamic or run-time) near8 (altering or changing or updating or configuring or reconfiguring) near8 (routing table)	36
☐	L9	L8 and (routing table)	101
☐	L8	L7 and @AD<20010816	4103
☐	L7	((partition or partitioning or configure or configuring) adj2 (algorithm or mechanism or software or scheme or program or policy or rule or regulation))	7867
☐	L6	L5 and (routing table)	1
☐	L5	L4 and @AD<20010816	290
☐	L4	(allocate or allocating or allocated or reallocate or reallocating or reallocated) near8 (processor or resources) and ((partition or partitioning) adj2 (algorithm or mechanism or software or scheme or program or policy or rule or regulation))	518
☐	L3	(allocate or allocating or allocated or reallocate or reallocating or reallocated) near8 (processor or resources) and ((partition or partitioning) adj2 (algorithm or mechanism or software or scheme or program))	491
☐	L2	(controller or manager) near8 (partition or partitioning) near8 (routing table)	3
☐	L1	(controller or manager) near8 (partition or partitioning) near8 (routing table) near8 ((multiple processors) or (multi-processors))	0

END OF SEARCH HISTORY

Previous Doc Next Doc Go to Doc#
First Hit

Generate Collection

L2: Entry 2 of 3 File: PGPB Dec 5, 2002

DOCUMENT-IDENTIFIER: US 20020184345 A1
TITLE: System and Method for partitioning a computer system into domains

Abstract Paragraph:
A domain partitioning system for a multi-node computer system is disclosed. An external server manager is coupled to a domain configuration unit by a dedicated sideband channel. The server manager has write privileges to the domain configuration unit that allows the server manager to control the domain partitioning and the routing tables. None of the domains of the computer system are permitted write access to the domain configuration unit. In one embodiment, the domain configuration unit is a set of domain partition registers and routing table registers coupled to a system interconnect.

Previous Doc Next Doc Go to Doc#

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L2: Entry 3 of 3 File: USPT Nov 1, 2005

DOCUMENT-IDENTIFIER: US 6961761 B2
TITLE: System and method for partitioning a computer system into domains

Abstract Text (1):
A domain partitioning system for a multi-node computer system is disclosed. An external server manager is coupled to a domain configuration unit by a dedicated sideband channel. The server manager has write privileges to the domain configuration unit that allows the server manager to control the domain partitioning and the routing tables. None of the domains of the computer system are permitted write access to the domain configuration unit. In one embodiment, the domain configuration unit is a set of domain partition registers and routing table registers coupled to a system interconnect.

Previous Doc Next Doc Go to Doc#

L16: Entry 6 of 15 File: USPT Jul 10, 2001

DOCUMENT-IDENTIFIER: US 6260068 B1
TITLE: Method and apparatus for migrating resources in a multi-processor computer system

Abstract Text (1):
Multiple instances of operating systems execute cooperatively in a single multi-processor computer wherein a single physical machine is subdivided by software into multiple partitions, each with the ability to run a distinct copy, or instance, of an operating system. Each individual instance has the resources it needs to execute independently, but instances cooperate to migrate resources from one partition to another. The migration can be initiated and carried out under control of the operating system instances "on the fly" without intervention of the system administrator. Alternatively, a system administrator can reconfigure the system. Resource migration is carried out under a "push" model in which resources are controlled by an owning partition and must be released by that partition before they can be migrated to another partition. In accordance with this model, a first operating system instance which requires a resource must first request the resource from a second instance. In response to this request, the second instance determines whether it can spare the resource, and if so, begins to bring the resource into an idle state. The resource is actually transferred when the second instance stops using the resource.

Application Filing Date (1):
19980610

Detailed Description Text (36):
The configuration tree 300 may extend to the level of device controllers, which will allow the operating system to build bus and device configuration tables without probing the buses. However, the tree may also end at any level, if all components below it cannot be configured independently. System software will still be required to probe for bus and device information not provided by the tree.

Detailed Description Text (128):
During initialization, the console program will modify the current_owner field to match the owner field for any node of which it is the owner, and for which the current_owner field is null. System software should only use hardware of which it is the current owner. In the case of a de-assignment of a resource which is owned by a community, it is the responsibility of system software to manage the transition between states. In some embodiments, a resource may be loaned to another partition. In this condition, the owner and current_owner fields are both valid, but not equal. The following table summarizes the possible resource states and the values of the owner and current_owner fields:

Detailed Description Paragraph Table (6):
TABLE 1 owner field value current_owner field value Resource State none none unowned, and inactive none valid unowned, but still active valid none owned, not yet active valid equal to owner owned and active valid is not equal to owner loaned

Previous Doc Next Doc Go to Doc#
First Hit Fwd Refs

Generate Collection

L16: Entry 12 of 15 File: USPT Aug 31, 1993

DOCUMENT-IDENTIFIER: US 5241652 A
TITLE: System for performing rule partitioning in a rete network

Abstract Text (1):
A rule-partitioning system for converting at least a portion of a target expert system program to a rule partitioned RETE network for execution on multiple processors, including a rule partitioning portion for assigning different rules of the target expert system program to different partitions on the basis of previously collected processing statistics and on the use of node sharing; and a compiler for converting the target expert system program to the RETE network, wherein the rules of the RETE network are assigned to the multiple processors in accordance with the partition assignments.

Application Filing Date (1):
19920331

Brief Summary Text (12):
The invention provides a new and improved system for rule partitioning in an expert system program to run on multiple processors.

Brief Summary Text (13):
In brief summary, in one aspect the invention is a rule-partitioning system for converting at least a portion of a target expert system program to a rule partitioned RETE network for execution on multiple processors. The invention includes a rule partitioning portion for assigning different rules of the target expert system program to different partitions on the basis of previously collected processing statistics and on the use of node sharing; and a compiler for converting the target expert system program to the RETE network, wherein at least some of the rules of the RETE network are assigned to the multiple processors in accordance with the partition assignments.

Brief Summary Text (15):
In another aspect, the invention is a system that includes a uniprocessor compiler for compiling a target system program to form a RETE network for execution on a single processor, some of the nodes of the uniprocessor RETE network possibly being shared by more than one rule; a processor for executing the target expert system program, the processor obtaining processing statistics in connection with each node of the uniprocessor RETE network during execution; a rule partitioning portion for assigning different rules of at least a portion of the target expert system program to different partitions based on the processing statistics and on the use of node sharing; and a multiprocessor compiler for converting the target expert system program to a RETE network for execution on multiple processors, wherein the rules of the multiprocessor RETE network are assigned to the multiple processors in accordance with the partition assignments.

Brief Summary Text (17):
In yet a further aspect, the invention is a computer program for use in connection with a computer. The computer program includes a uniprocessor compiler for enabling the computer to compile a target system program to form a RETE network for

execution on a single processor, some of the nodes of the uniprocessor RETE network possibly being shared by more than rule; a module for enabling the computer to execute the target expert system program, the computer obtaining processing statistics in connection with each node of the uniprocessor RETE network during execution; a rule partitioning module for enabling the computer to assign different rules of at least a portion of the target expert system program to different partitions based on the processing statistics and on the use of node sharing; and a multiprocessor compiler module for enabling the computer to convert the target expert system program to a RETE network for execution on multiple processors, wherein the rules of the multiprocessor RETE network are assigned to the multiple processors in accordance with the partition assignments.

Detailed Description Text (69):
Also, during the compilation step, a mapping that identifies the join nodes 112 corresponding to each particular rule of all the rules to be rule partitioned is generated and stored in a lookaside table 160, such as the one illustrated in FIG. 5. Only the rules deemed unsuitable for beta partitioning (see step 102 of FIG. 3A) are included in the group of rules to be rule partitioned. Lookaside table 160 includes a sufficient number of blocks 162 to represent all of the rules in the target expert system program that are being rule partitioned. Each block 162 includes a rule identifier field 164, a mapping field 166, a total time field 168 and a partition identifier field 170. Each rule for which a mapping is generated is identified in rule identifier field 164 and a list identifying the join nodes which make up that rule is stored in the mapping field 166.

Detailed Description Text (70):
In the described embodiment, the processor accomplishes this mapping by first grouping join nodes 112 into what shall be referred to as .mu.-blocks 114 and then mapping the .mu.-blocks to the rules (that is, identifying, for each rule its component .mu.-blocks) in lookaside table 160. Each .mu.-block 114 is the longest sequence of connected join nodes 112 that can be constructed without violating the rule that all nodes of any given .mu.-block 114 must be shared by all members of a set of rules. Thus, in FIG. 4B for example, join nodes 112 numbered 1 and 2 are members of the same .mu.-block 114, namely, .mu..sub.1, because they are part of a sequence and they are each shared by all members of a set of rules, namely, Rules A, B and C. Join node 112 numbered 3, however, cannot be a member of .mu..sub.1 because it is not shared by the same set of rules (it is shared only by Rules A and C). All join nodes 112 are assigned to a corresponding numbered .mu.-block 114 in accordance with these criteria. FIG. 4B depicts a set of .mu.-blocks 114, namely, .mu..sub.1 through .mu..sub.5, which are constructed in accordance with such rules.

Detailed Description Text (83):
After the S.sub.j 's are computed for all of .mu.-blocks 114, the processor computes the total time spent in each rule for all of the cycles of the program and stores this in the appropriate total time field 168 in lookaside table 160. The total time for a rule, T.sub.rule, is the sum of the total times for the .mu.-blocks 114 which make up the rule. For example, as shown in FIG. 5, Rule A consists of three .mu.-blocks 114, namely, .mu..sub.1, .mu..sub.2, and .mu..sub.3 ; thus, T.sub.A =S.sub.1 +S.sub.2 +S.sub.3.

Detailed Description Text (84):
The statistics stored in tables 160 and 172 are subsequently used by a rule partition system (referred to hereinafter as the RP system) to distribute the processing of the rules among the processors of a multiprocessor system by assigning different rules to different processors (see step 104 of FIG. 3A). As shown in FIGS. 7A and 7B, the RP system initially rearranges the order of the rules listed in lookaside table 160 so that they are in descending order according to T.sub.rule, that is, the total time spent on processing each rule, with the most time consuming rule (i.e., the rule associated with the block 162 having the

highest value in its total time field 168) being listed first in table 160 (step 200). Then, the RP system generates a group of empty partition sets, .pi..sub.n 1.ltoreq.n.ltoreq.N, where N is specified by an operator and equals the number of central processing units (CPU's) in the multiprocessor system for which the target expert system program is being rule partitioned (step 210).

Detailed Description Text (85):
Briefly, after the partition sets have been generated, the RP system, for each of the target expert system rules identified in table 160, iteratively performs a series of operations to determine a partition to which the rule should be permanently assigned. During each iteration, the RP system selects a rule and temporarily assigns it to a partition and determines processing times for all of the partitions.

Detailed Description Text (87):
The RP system iteratively performs these operations to determine the total processing time for each temporary assignment of the selected rule to each partition. Thereafter, it determines the partition having the lowest total processing time, and permanently assigns the rule to that partition. The RP system repeats these iterative operations for each rule identified in table 160, until all of the rules have been assigned to partitions. By these operations, the RP system can determine the assignments of the rules to the various partitions to provide near optimum total processing time for all the rules in the target expert system program that are to undergo rule partitioning.

Detailed Description Text (88):
More specifically, the RP system initially selects the most time consuming rule among the unassigned rules listed in lookaside table 160 (i.e., the rule with the largest T.sub.rule) (step 220). During processing, the RP system iteratively selects the rules in the order they are listed in the table. After selecting a rule, the RP system initializes an index, k, by setting it equal to one (step 230). The index is used to identify the particular partition .pi..sub.k to which the rule is being assigned, and also to determine when the rule has been assigned to all partitions, at which point iterations for that rule stop. Next, after initializing the index k, the RP system temporarily assigns the selected rule to partition .pi..sub.k, i.e., the partition identified by the index k (step 240). Based upon this temporary assignment, the RP system computes, for each partition and for each cycle of the program, the time, as determined from array 172, for processing the rules (including the temporarily-assigned rule as well as rules, if any, which were previously permanently assigned to the partition) assigned to the respective partitions (step 250). These process times are derived from the information stored in lookaside table 160, which identifies the .mu.-blocks assigned to each rule of the target expert system program, and effect data array 172, which lists the time to process each .mu.-block for each cycle of the target expert system program.

Detailed Description Text (97):
By the time the RP system has sequenced to stop 300 it has determined total program processing times for the temporary assignment of the selected rule to the various partitions. In step 300, the RP system identifies the partition assignment of the selected rule that yielded the lowest total program processing time and permanently assigns the selected rule to that partition. The permanent assignment of the rule to a partition is then recorded in the corresponding partition assignment field 170 of lookaside table 160 (see FIG. 5) of block 162 for the rule. If more than one temporary partition assignment .pi..sub.k for the selected rule have the same lowest total program processing time, as will be the case for the first rule in table 160, the RP system assigns the selected rule to the partition .pi..sub.k having both the lowest total program processing time and the lowest index number, k. Since Rule A is the first rule to be selected and no other rules have yet been permanently assigned to any partition .pi..sub.k, the RP system will determine the

same processing time for both partition .pi..sub.1 and partition .pi..sub.2. Thus, Rule A will be assigned to partition .pi..sub.1.

Detailed Description Text (98):
After the first selected rule is assigned to a partition, the RP system determines whether there are any other rules which have not been assigned to partitions .pi..sub.k (step 310). If the RP system determines that there are no other rules which have not been assigned, it exits and the rule partitioning is completed. However, if table 160 includes other rules which have not been assigned to a partition .pi..sub.k, the RP system branches back to step 220 in which it selects the next rule, which is the next most time consuming unassigned rule and repeats the above-described steps to determine a permanent assignment for that rule.

Detailed Description Text (99):
In determining the processing times for the temporary assignments in each iteration (step 250), the RP system takes into account when nodes are shared among rules. This will be illustrated by the following discussion of the processing of Rule B by the RP system. For example, continuing with arrangement depicted in FIG. 5, if it is assumed that Rule B is the second most time consuming rule in table 160, then during the first iteration of processing Rule B, Rule B is first temporarily assigned to .pi..sub.1, to which, as described above, Rule A has been permanently assigned. Given this temporary assignment, the computed process times associated with each partition for the first three cycles of the program are:

Detailed Description Text (111):
Finally, in step 300, Rule C is permanently assigned to the partition which yields the smallest TOT(x.fwdarw.k). These operations are repeated for each of the rules in table 160.

CLAIMS:

16. A system for building a reticular discrimination network (RETE network) capable of being partitioned across a set of multiple, interconnected computer processors that comprise a computer system, from at least a portion of a target expert system program, the target expert system program comprising a plurality of rules, each of which comprise a plurality of conditions grouped in a sequence by a set of logical connectors and a result to occur when the conditions of the rule are satisfied, certain ones of the plurality of rules having conditions that are equivalent to conditions in other ones of the plurality of rules, each of the conditions in a rule being represented by a test node in the RETE network, the test nodes being connected through associated join nodes representing the logical connectors of the rule, the system for building a RETE network comprising:

a. a memory configured to store the target expert system program;

b. a uniprocessor compiler coupled to the memory and configured to compile the target expert system program to form a RETE network for execution on a single processor and store the compiled target expert system program in the memory, certain ones of the conditions represented by nodes of the uniprocessor RETE network being shared by more than one rule;

c. a processor, coupled to the memory for executing the target expert system program, to obtain processing statistics in connection with each node of the uniprocessor RETE network during execution;

d. a partitioner, coupled to the memory and configured to read each condition of each rule of at least a portion of the target expert system program, and partition by associating each rule into one of a preselected set of partitions, and store partition information relating to the partitioning in the memory, each partition

representing one of the multiple processors comprising the computer system, the partitioning performed in a manner to identify conditions shared by more than one rule and partition the rules as a function of the identified shared conditions and the processing statistics; and

e. a compiler, coupled to the memory and configured to convert the rules of the at least portion of the target expert system program into a RETE network, each partitioned rule being compiled to execute on a corresponding processor based on the partition information.

31. A method for building a reticular discrimination network (RETE network) capable of being partitioned across a set of multiple, interconnected computer processors that comprise a computer system, from at least a portion of a target expert system program, the target expert system program comprising a plurality rules each of which comprise a plurality of conditions, grouped in a sequence by a set of logical connectors, and a result to occur when the conditions of the rule are satisfied, certain ones of the plurality of rules having conditions that are equivalent to conditions in other ones of the plurality of rules, each of the conditions in the a rule being represented by a test node in the reticular discrimination net, the test nodes being connected through associated join nodes representing the logical connectors of the rule, the method comprising the steps of:

a. storing the target expert system program in memory;

b. compiling the target expert system program to form a RETE network for execution on a single processor;

c. executing the target expert system program on a single processor;

d. obtaining processing statistics in connection with each node of the RETE network during execution of the target expert system program;

e. partitioning each rule of at least a portion of the target expert system program by reading each condition of each rule from the memory and associating each rule into one of a preselected set of partitions, each partition representing one of the multiple processors comprising the computer system, the partitioning performed in a manner to identify conditions shared by more than one rule and partition the rules as a function of the identified shared conditions and the processing statistics;

f. storing partition information relating to the partitioning in the memory; and

g. compiling the at least a portion of the target expert system program by converting the rules of the target expert system program into a RETE network, each partitioned rule being compiled to execute on a corresponding processor based on the partitioning information.

44. A method for building a reticular discrimination network (RETE network) capable of being partitioned across a set of multiple, interconnected computer processors that comprise a computer system, from at least a portion of a target expert system, the target expert system comprising a plurality of rules each of which comprise a plurality of conditions, grouped in a sequence by a set of logical connectors, and a result to occur when the conditions of the rule are satisfied, certain ones of the plurality of rules having conditions that are equivalent to conditions in other ones of the plurality of rules, each of the conditions in a rule being represented by a test node in the reticular discrimination network, the test nodes being connected through associated join nodes representing the logical connectors of the rule, the method comprising:

a. storing the target expert system in memory;

b. partitioning each rule of the at least a portion of the target expert system by associating each rule into one of a preselected set of partitions, each partition representing one of the multiple processors comprising the computer system, the partitioning step being performed in a manner to identify conditions shared by more than one rule, the partitioning step being further performed in manner to increase the likelihood that a first and second rule will be associated into a same one of the preselected set of partitions when the first and second rules share any of certain ones of the plurality of conditions;

c. storing the partition information in the memory; and

d. compiling the at least a portion of the target expert system in the memory to convert the rules of the target expert system into a RETE network, each partitioned rule being compiled to execute on a corresponding microprocessor based on the partition information.

Previous Doc Next Doc Go to Doc#